OFFICE OF CIVILIAN RADIOACTIVE WASTE MANAGEMENT
CALCULATION COVER SHEET

1. QA: QA

Page: 1 Of: 6

2. Calculation Title

Testing of Software Routine to Determine Deviate and Cumulative Probability: ModStandardNormal Version 1.0

3. Document Identifier (including Revision Number)

CAL-EBS-MD-000004 REV 00

4. Total Attachments	5. Attachment Numbers - Number of pages in each
4	I-3, II-20, III-4, IV-2

	Print Name	Signature	Date
6. Originator	Ahmed M. Monib	Ahmed M Monib	10/5/99
7. Checker	Nelson J. Erb	Nelson J Erb	10/5/99
8. Lead	Joon H. Lee	Joon H Lee	10/5/99

9. Remarks

Revision History

10. Revision No.	11. Description of Revision
REV 00	Initial Issue

CONTENTS

1. PURPOSE

The purpose of this calculation is to document that the software routine *ModStandardNormal Version 1.0* which is a Visual Fortran 5.0 module, provides correct results for a normal distribution up to five significant figures (three significant figures at the function tails) for a specified range of input parameters. The software routine may be used for quality affecting work. Two types of output are generated in *ModStandardNormal*: a deviate, x, given a cumulative probability, p, between 0 and 1; and a cumulative probability, p, given a deviate, x, between –8 and 8. This calculation supports Performance Assessment, under Technical Product Development Plan, TDP-EBS-MD-000006 (Attachment I, DIRS 3) and is written in accordance with the AP-3.12Q *Calculations* procedure (Attachment I, DIRS 4).

2. METHOD

The *ModStandardNormal* routine performs the two functions stated above using three test driver programs. The first driver program, *testfwdnorm*, uses the cumulative probability function to numerically estimate the probability, p, given the deviate x. The procedure is referred to as the forward normal calculation. The second driver program, *testinvnorm*, uses the cumulative probability function to numerically estimate the deviate, x, given the cumulative probability, p and the procedure is referred to as the inverse normal calculation. The third test driver program, *testnormtable,* interopolates the deviate and cumulative probability from a table. The numerical functions and the table used can be found in Attachment II. A more detailed explanation on calculating the forward and inverse normals can be found in select journal articles (Attachment I, DIRS 1 and 2).

The *ModStandardNormal* routine generates the normal distribution function by evaluating the error function as shown in Equation 1. Since the error function is only defined for deviate values between 0 and + ∞, and since the normal distribution function is symmetrical about the y-axis, probability values for negative deviates are obtained by evaluating 1-*erf (x)*. To evaluate the integral numerically, a Chebyshev expansion, which is essentially a Fourier expansion, is used (Attachment I, DIRS 1).

$$erf(x) = \frac{2}{\pi} \int_0^x e^{-t^2} dx \qquad \text{(Eq. 1)}$$

ModStandardNorm uses the Chebyshev expansion to generate a normal table of 4900 equally sized increments of size 0.0001 for probabilities $0.01 < p < 0.50$. Cumulative probability values of $0.50 < p < 1.0$ are calculated by recognizing that the standard normal curve is symmetrical. Thus $1\text{-}p = p$ for p between $0.50 < p < 1.0$. For values within the range but not in the table, *ModStandardNorm* uses linear interpolation to determine the deviate (or cumulative probability). For the small number of cumulative probabilities that fall outside the range $0.01 < p < 0.99$, *ModStandardNorm* uses the Chebyshev expansion to determine each deviate (or cumulative probability) value.

In order to document that *ModStandardNormal* provides correct results, the three driver programs used in *ModStandardNormal* are compared with the built-in cumulative probability functions in the MathCad software. Sixteen deviate values between –8 and 8 were calculated using both the built-in MathCad function *qnorm,* and the Fortran driver program, *testinvnorm,* and their absolute error was determined. Twenty-three cumulative probability values between 1×10^{-15} and $9.99999999999999 \times 10^{-1}$ were also calculated using both the built-in MathCad function *pnorm*, and the Fortran driver program, *testfwdnorm*, and their absolute error determined. The absolute error from the output of the two MathCad functions were also evaluated in relation to the Fortran driver program, *testnormtable*, which interpolates the deviate and cumulative probability from a table.

3. ASSUMPTIONS

The following assumption was made in performing this calculation:
To test the consistency between the *ModStandardNormal* subroutine and the MathCad worksheets, it was assumed sufficient to have agreement up to five significant figures. Thus, an absolute error equal to or less than 1×10^{-5} is taken to be a reasonable level of accuracy. However, for probabilities less than or equal to 1×10^{-15} and greater than $9.99999999999999 \times 10^{-1}$, an absolute error equal to or less than 1×10^{-3} is taken to be reasonable. Given that MathCad and the Fortran routine use slightly different polynomial expansions to numerically compute the cumulative distribution function, greater accuracy is not possible, especially at the tails of the distribution. This assumption is used throughout.

4. USE OF COMPUTER SOFTWARE AND MODELS

Appropriate industry standard software used in this calculation are MathCad Professional Version 8.02 and Excel 97 SR-2. This software was used to perform "hand calculation" verification of Visual Fortran 5.0 routines. Visual Fortran 5.0 is also industry standard software. Both software programs were executed on an IBM-compatible, DELL PowerEdge 2200 Workstation equipped with a Pentium II 266 MHz processor (CRWMS M&O tag 111593) in the Windows NT 4.0 operating system. Details of the MathCad worksheets used and the Fortran routines can be found in Attachment III (MathCad Worksheet) and Attachment II (Fortran Routines), respectively. A description of the governing formulas used in the ModStandardNormal routine was described in Section 2.

5. CALCULATION

A MathCad worksheet was written to duplicate the three Fortran driver programs used to run the *ModStandardNorm* subroutines. The driver program, *testfwdnorm* was emulated using the built-in MathCad function *pnorm*. The driver program, *testinvnorm* was emulated using the built-in MathCad function *qnorm*. The driver program *testnormtable* was emulated using both MathCad functions *pnorm and qnorm*. *Testnormtable* interpolates either the deviate or cumulative probability from a table. The Fortran driver programs and *ModStandardNormal* routines can be found in Attachment II and the MathCad worksheet is shown in Attachment III.

To test *testfwdnorm, ModStandNorm* and the *tesfwdnorm* driver program were compiled and executed (a general-purpose module, *moddefaultsize*, is first compiled to define default byte size for reals, integers and logicals). Upon execution, the user is prompted to enter a deviate between –8 and 8. *ModStandNorm* then generates the corresponding cumulative probability value and prompts the user to enter another deviate between –8 and 8. To exit the program the user enters a value outside of the specified range. The input and output values are exported to the file, *TestFwdNorm.out* and the results are then copied into a column of an embedded Excel spreadsheet in the *StandNorm* MathCad worksheet. The same deviate values are used in the MathCad worksheet's built-in function *pnorm* and the results saved to the preceding column of the embedded Excel sheet. The relative error between the two columns of numbers was then calculated.

The same procedure described above for the *testfwdnorm* driver and *pnorm* function is used for the *testinvnorm* driver and *qnorm* function, and also the *testnormtable* driver and *pnorm/qnorm* functions. The Fortran input and output values for *testinvnorm* were exported to the file *TestInvNor.out*, and the input and ouput values for *testnormtable* were exported to the file *TestNormTable.out*. The Fortran output files are shown in Attachment IV. Since the inputs used in this calculation are not considered data, they do not require a TBV/TBD tracking number.

6. RESULTS

All input and output information relevant to this calculation are included in Attachments III and IV. For brevity, only a select portion is reproduced in hardcopy form within this section. This calculation does not contain information or assumptions that need to be confirmed prior to the use of the results of this calculation.

Shown below, is the embedded Excel worksheet within MathCad for the results of the inverse normal calculations (given a probability, p, find the deviate, x).

The input probability is shown in column A and the output of the built-in MathCad function *qnorm* is shown in column B. The Fortran outputs for *testnormtable* and *testinvnorm* are shown in columns C and D, respectively. Column E shows the absolute relative error between the MathCad output and *testnormtable*, while column F shows the absolute error between the MathCad output and *testinvnorm*. The absolute error for all input values are below 1×10^{-5}, with the exceptions of probabilities less than or equal to 1×10^{-15} and greater than $9.99999999999999 \times 10^{-1}$, in which case the absolute error values are below 1×10^{-3}. The same is true for the forward normal calculations (given a deviate x, find the cumulative probability p) and the results are found in Attachment III.

A	B	C	D	E	F
Cum Prob. (p)	MathCad (x)	NormTable (x)	Function (x)	Absolute Error |(B-C)|	Absolute Error |(B-D)|
1.0000E-15	-7.9414E+00	**N/A**	-7.9420E+00	**N/A**	5.9444E-04
2.0000E-03	-2.8782E+00	**N/A**	-2.8782E+00	**N/A**	4.9370E-08
4.0000E-03	-2.6521E+00	**N/A**	-2.6521E+00	**N/A**	4.5278E-08
8.0000E-03	-2.4089E+00	**N/A**	-2.4089E+00	**N/A**	4.1107E-08
1.0000E-02	-2.3263E+00	-2.3263E+00	-2.3263E+00	3.9805E-08	3.9805E-08
1.0050E-02	-2.3245E+00	-2.3245E+00	-2.3245E+00	4.0153E-06	3.9775E-08
1.0070E-02	-2.3237E+00	-2.3237E+00	-2.3237E+00	3.3622E-06	3.9764E-08
1.0090E-02	-2.3230E+00	-2.3230E+00	-2.3230E+00	1.4164E-06	3.9752E-08
2.0000E-02	-2.0537E+00	-2.0537E+00	-2.0537E+00	3.5281E-08	3.5281E-08
6.0000E-02	-1.5548E+00	-1.5548E+00	-1.5548E+00	2.6589E-08	2.6589E-08
1.0000E-01	-1.2816E+00	-1.2816E+00	-1.2816E+00	2.1930E-08	2.1930E-08
2.0000E-01	-8.4162E-01	-8.4162E-01	-8.4162E-01	1.4393E-08	1.4393E-08
4.0000E-01	-2.5335E-01	-2.5335E-01	-2.5335E-01	4.3358E-09	4.3359E-09
5.0000E-01	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
7.0000E-01	5.2440E-01	5.2440E-01	5.2440E-01	8.9747E-09	8.9747E-09
9.999999999999990E-01	7.94100472	**N/A**	7.94E+00	**N/A**	4.3963E-04

7. ATTACHMENTS

Attachment Number	Title
I	DIRS
II	Fortran Routines
III	MathCad Worksheet For Generating Random Numbers
IV	Fortran Output Files

ATTACHMENT I- Document Input Reference Sheet (DIRS)

OFFICE OF CIVILIAN RADIOACTIVE WASTE MANAGEMENT

DOCUMENT INPUT REFERENCE SHEET

1. Document Identifier No./Rev.: CAL-EBS-MD-000004 REV 00	Change:	Title: Testing of Software Routine to Determine Deviate and Cumulative Probability: ModStandardNormal Version 1.0

Input Document			4. Input Status	5. Section Used in	6. Input Description	7. TBV/TBD Priority	8. TBV Due To		
2. Technical Product Input Source Title and Identifier(s) with Version		3. Section					Unqual.	From Uncontrolled Source	Unconfirmed
2a 1	Shampine, L.F 1973. "Exact Solutions for Concentration Dependent Diffusion and the Inverse Complimentary Error Function." *Journal of The Franklin Institute*, 295, 239-247. New York, New York: Pergamon Press. TIC: 243310	Entire	N/A	Section 2	Polynomial Approximations for the Inverse Normal	N/A	N/A	N/A	N/A
2	Amos, D.L. and Daniel, S.L. 1972. "CDC 6600 Codes for the Error Function, Cumulative Normal and Related Functions." SC-DR-72-0918. Albuquerque, New Mexico. TIC: 244294	Entire	N/A	Section 2	Polynomial Approximations For the Forward Normal	N/A	N/A	N/A	N/A

ATTACHMENT I- Document Input Reference Sheet (DIRS)

OFFICE OF CIVILIAN RADIOACTIVE WASTE MANAGEMENT

DOCUMENT INPUT REFERENCE SHEET

1. Document Identifier No./Rev.: CAL-EBS-MD-000004 REV 00			Change:	Title: Testing of Software Routine to Determine Deviate and Cumulative Probability: ModStandardNormal Version 1.0					
Input Document			4. Input Status	5. Section Used in	6. Input Description	7. TBV/TBD Priority	8. TBV Due To		
2. Technical Product Input Source Title and Identifier(s) with Version		3. Section					Unqual.	From Uncontrolled Source	Unconfirmed
3	CRWMS M&O 1999. *Testing of Software Routines Applicable to EBS Performance Assessment.* TDP-EBS-MD-000006 REV 00. Las Vegas, Nevada: CRWMS M&O. ACC: MOL.19990811.0073	Entire	N/A	Entire	Plan under which Calculation is being performed	N/A	N/A	N/A	N/A
4	AP-3.12Q, Rev. 0. *Calculations.* Washington, D.C.: U.S. Department of Energy, Office of Civilian Radioactive Waste management. ACC: MOL.19990702.0312	Entire	N/A	Entire	Procedure used in writing document	N/A	N/A	N/A	N/A

ATTACHMENT I- Document Input Reference Sheet (DIRS)

OFFICE OF CIVILIAN RADIOACTIVE WASTE MANAGEMENT

DOCUMENT INPUT REFERENCE SHEET

1. Document Identifier No./Rev.: CAL-EBS-MD-000004 REV 00			Change:	Title: Testing of Software Routine to Determine Deviate and Cumulative Probability: ModStandardNormal Version 1.0					
Input Document			4. Input Status	5. Section Used in	6. Input Description	7. TBV/TBD Priority	8. TBV Due To		
2. Technical Product Input Source Title and Identifier(s) with Version		3. Section					Unqual.	From Uncontrolled Source	Unconfirmed
	AP-3.12Q, Rev. 0. *Calculations.* Washington, D.C.: U.S. Department of Energy, Office of Civilian Radioactive Waste management. ACC: MOL.19990702.0312								

AP-3.15Q.1 Rev. 06/30/1999

ATTACHMENT II – FORTRAN ROUTINES

CONTENTS

1. MODULE: MODDEFAULTSIZE

```
MODULE moddefaultsize
c
c  This module is general purpose to define default byte size for reals,
c  integers, and logicals.  Consistent use throughout a program will
c  ensure argument list conformity, etc.
c
    implicit none
    integer(1), PARAMETER :: RKind = 8
    integer(1), PARAMETER :: IKind = 4
    integer(1), PARAMETER :: LKind = 1

    END MODULE moddefaultsize
```

2. MODSTANDARDNORMAL

```
MODULE modstandardnormal
c
c  The purpose of this module is to provide p given x or x given p in
c  the expression p = pr(z < x) where pr denotes probability and z
c  indicates the standard normal distribution.  Which routines to call
c  depends on the number of calls that will be made.
c  1. If fewer than about 5000 calls are to be made, the most efficient
c     way is: use InvNor(given p, find x) and FwdNorm(given x, find p).
c  2. If more than about 5000 calls are to be made, the most efficient
c     way is to use StdNorDev(given p, find x) and StdNorProb (given x,
c     find p).
c  For the latter, tables of deviates are generated in a manner that the
c  associated cumulative probabilites can be ascertained from their
c  indexes.  Then when StdNorDev is called with a probability argument,
c  deviate x is interpolated from the values in the tables.  Similarly,
c  when StdNorProb is called with a deviate argument, probability p is
c  interpolated from the table indexes.  To generate the table, array v
c  is allocated.  If there is not sufficient memory for the allocation,
c  routines InvNor and FwdNorm will be used.
c
    USE moddefaultsize
    Public  StdNorDev, StdNorProb, InvNor, FwdNorm
    Private MakeNormTable, InvERFC, BiSearchOnv, Erf,
   &        ErfAbsLT2, Erf2To4, Erf4To6, ErfChebyPoly
c
    real(RKind), dimension(:), allocatable, save, private :: v
    real(RKind), save, private :: MinProb4v, MaxProb4v
    logical(LKind), save, private :: TableMade
```

```
c
      data TableMade / .false. /
c
      CONTAINS    !MakeNormTable, StdNorDev, StdNorProb
                  !InvNor, InvERFC, BiSearchOnv, FwdNorm, Erf
                  !ErfAbsLT2, Erf2To4, Erf4To6, ErfChebyPoly
c
c**********************************************************************
c
      SUBROUTINE MakeNormTable
c
c  Make a normal table for the interior of the standard normal.  The
c  'interior' for table v accounts for probabilities p such that
c  0.01 < p < 0.5 at 4900 equal increments of size 0.0001.  Value v(i)
c  is the standard normal deviate at the cumulative probability
c  (i + 100) / 10,000 for indexes i such that 0 <= i <= 4900.
c  Input :(none)
c  Output:(none through the argument list, although v, MinProb4v,
c         MaxProb4v, and TableMade are defined at module-level)
c  Local :vDim, i, AllocStat, pr, ProbStep4v
c
c  Local variables
c
      integer(IKind) :: vDim, i, AllocStat
      real(RKind) :: pr, ProbStep4v
c
      vDim      = 4900
      MinProb4v  = 0.01_RKind
      MaxProb4v  = 0.99_RKind
      ProbStep4v = 0.0001_RKind
c
c  If insufficient memory to store v, return before setting TableMade
c  to true.
c
      allocate(v(0: vDim), stat = AllocStat)
      if(AllocStat .ne. 0) RETURN
c
      pr     = MinProb4v
      v(0)   = InvNor(pr)
      v(vDim) = 0.0_RKind
      do i = 1, vDim - 1
        pr   = pr + ProbStep4v
        v(i) = InvNor(pr)
      end do
c
      TableMade = .true.
```

```
    RETURN
    END SUBROUTINE MakeNormTable
c
c*******************************************************************************
c
    real(RKind) function StdNorDev(parg)
c
c  Interpolation routine for a NORMAL lookup table.  Argument parg is a
c  cumulative probability.  The corresponding standard normal deviate is
c  found and returned as the function value.  The value of v(i) is the
c  standard normal deviate at cumulative probability (i + 100) / 10,000
c  where 0 <= i <= 4900.  For probabilities outside 0.01 to 0.99,
c  function InvNor is used to compute the deviate.
c
c  Input : parg
c  Output: (function value)
c  Local : KI, p, x, y, Rat
c
c  Arguments
c
    real(RKind) :: parg
c
c  Local variables
c
    integer(IKind) :: KI
    real(RKind) :: p, x, y, Rat
c
c  If the normal table is not yet made, do so now.
c
    if(.not. TableMade) then
     CALL MakeNormTable
     if(.not. TableMade) then     !Insufficient memory
      x = InvNor(p)
      StdNorDev = x
      RETURN
     end if
    end if
c
c  If the probability is on the tail, find the deviate directly using
c  function InvNor.  Otherwise, linearly interpolate from the table,
c  taking advantage of symmetry.
c
    p = parg
    if(p .le. MinProb4v .or. p .ge. MaxProb4v) then
     x       = InvNor(p)
     StdNorDev = x
```

```
c
    else if(p .gt. 0.5_RKind) then
      p       = 10000.0_RKind * (1.0_RKind - p)
      Rat     = p - int(p)
      KI      = int(p - 100.0_RKind)
      y       = Rat * (v(KI + 1) - v(KI)) + v(KI)
      StdNorDev = -y
c
    else if(p .lt. 0.5_RKind) then
      p       = 10000.0_RKind * p
      Rat     = p - int(p)
      KI      = int(p - 100.0_RKind)
      y       = Rat * (v(KI + 1) - v(KI)) + v(KI)
      StdNorDev = y
c
    else
      StdNorDev = 0.0_RKind
    end if
    RETURN
    END FUNCTION StdNorDev
c
c**********************************************************************
c
    real(RKind) function StdNorProb(xarg)
c
c  Argument xarg is a deviate from the standard normal distribution.
c  Find the cumulative probability and return as the function value.
c  The value of v(i) is the standard normal deviate at cumulative
c  probability (i + 100) / 10,000, where 0 <= i <= 4900. For probability
c  outside 0.01 to 0.99, function FwdNorm is used.
c  Input : xarg
c  Output: (function value)
c  Local : CompProb, KI, FirstNdx, LastNdx, p, Rat, x
c
c  Arguments
c
    real(RKind) :: xarg
c
c  Local variables
c
    logical(LKind) :: CompProb
    integer(IKind) :: KI, FirstNdx, LastNdx
    real(RKind) :: p, Rat, x
c
c  If the normal tables are not yet made, do so now.
c
```

```
    if(.not. TableMade) then
     CALL MakeNormTable
     if(.not. TableMade) then          !Insufficient memory
      if(abs(xarg) .gt. 0.0_RKind) then
       p = FwdNorm(xarg)
       StdNorProb = p
      else
       StdNorProb = 0.5_RKind
      end if
      RETURN
     end if
    end if
c
c  For argument xarg, there are four tests of its value:
c  1. If xarg is such that the resulting probability p will be outside
c     the range from 1E-15 to 0.999999999999999, then either p = 0 or 1
c     will be returned, as a limiting value.
c  2. ElseIf xarg is such that p will be inside the range of (1) but
c     outside the range from 0.01 to 0.99, then function FwdNorm is
c     called to find the probability.
c  3. ElseIf xarg is such that p will be inside the range from 0.01 to
c     0.99 (but xarg is not zero), then find the stored entries in v
c     that surround xarg and linearly interpolate. The search for the
c     values that surround xarg uses bisection (BiSearchOnv).
c  4. Else xarg must be exact zero so that p = 0.5.
c
    x = xarg
    if(abs(x) .gt. 7.9414_RKind) then          ! Case 1
     if(x .gt. 0.0_RKind) then
      StdNorProb = 1.0_RKind
     else
      StdNorProb = 0.0_RKind
     end if
c
    elseif(abs(x) .ge. abs(v(0))) then         ! Case 2
     p  = FwdNorm(x)
     StdNorProb = p
c
    else if(abs(x) .gt. 0.0_RKind) then        ! Case 3
     if(x .gt. 0.0_RKind) then
      CompProb = .true.    !Find complementary probability first
      x = -x
     else
      CompProb = .false.
     end if
     FirstNdx = 0
```

```
      LastNdx  = 4900
      KI  = BiSearchOnv(x, FirstNdx, LastNdx)
      Rat = (x - v(KI - 1)) / (v(KI) - v(KI - 1))
      p   = ((KI - 1) + 100.0_RKind + Rat) * 0.0001_RKind
      if(CompProb) p = 1.0_RKind - p
      StdNorProb = p
c
    else                              ! Case 4
      StdNorProb = 0.5_RKind
    end if
    RETURN
    END FUNCTION StdNorProb
c
c*******************************************************************************
c
    real(RKind) FUNCTION InvNor(p)
c
c  Given cumulative probability p, find deviate x from the standard
c  normal distribution such that p = pr(z < x).  Use the numerical
c  approximation to the complementary error function found in function
c  InvERFC.  Ensure the argument for InvERFC is valid.
c  Input : p
c  Output: function value of InvNor
c  Local : sqrt2
c
c  Argument
c
    real(RKind) :: p
c
c  Local variable
c
    real(RKind) :: sqrt2
    save sqrt2
    data sqrt2 / 1.4142135623731 /
c
c  Although function InvErfc is valid for arguments less than 1E-15 (and
c  greater than 6.63967719958073E-36), restrict the valid lower bound on
c  probability to 1E-15 to maintain symmetry with the largest possible
c  probability, ie, 1 - 1E-15.
c
    if (p .lt. 1.0E-15) then
      InvNor = -7.942_RKind
    else if (p .lt. 0.5_RKind) then
      InvNor = -sqrt2 * InvERFC(2.0_RKind * p)
    else if (p .gt. 0.999999999999999_RKind) then
      InvNor = 7.942_RKind
```

```
      else if (p .gt. 0.5_RKind) then
        InvNor = sqrt2 * InvERFC(2.0_RKind * (1.0_RKind - p))
      else
        InvNor = 0.0_RKind
      end if
      RETURN
      END FUNCTION InvNor
c
c*********************************************************************
c
      real(RKind) FUNCTION InvERFC(Y)
c
c  Approximate the inverse complementary error function at Y. Valid for
c  argument Y greater than 6.63967719958073D-36 and less than 1.D0.
c  For best results, byte-size parameter RKind should be at least 8.
c  Chebyshev polynomials due to Shampine.
c  Input : Y
c  Output: function value
c  Local : I, J, K, D, TD, W, VN, VNP1, TEMP, A, A1, A2, A3, C1, C2
c
c  Argument
c
      real(RKind) :: Y
c
c  Local variables
c
      real(RKind) :: A(22, 3), A1(22), A2(22), A3(22), C1, C2
      EQUIVALENCE (A(1, 1), A1(1))
      EQUIVALENCE (A(1, 2), A2(1))
      EQUIVALENCE (A(1, 3), A3(1))
      integer(IKind) :: I, J, K
      real(RKind) :: D, TD, W, VN, VNP1, TEMP
      save A, A1, A2, A3, C1, C2
c
      DATA(A1(I), I = 1, 22) /  9.18725611735013D-01, 0.,
     & 1.68792878000327D-02, 0., 6.60337139058300D-04, 0.,
     & 3.20203849839380D-05, 0., 1.72060607522481D-06, 0.,
     & 9.81965971588191D-08, 0., 5.83049613537653D-09, 0.,
     & 3.56019351836136D-10, 0., 2.21968915783128D-11, 0.,
     & 1.40639693109741D-12, 0., 9.02597345404862D-14, 0./
c
      DATA(A2(I), I=1,22) /  1.54701109458613D+00,-3.31460331083896D-01,
     & 4.33001124090060D-02,-1.06564004165532D-02, 2.90613542304156D-03,
     &-8.61872838022491D-04, 2.67933751795053D-04,-8.60838893942933D-05,
     & 2.83232058814598D-05,-9.48870819734494D-06, 3.22422655069385D-06,
     &-1.10815778472076D-06, 3.84464770797987D-07,-1.34439275565208D-07,
```

```
     & 4.73255976052393D-08,-1.67556011100019D-08, 5.96199003969093D-09,
     &-2.13070503291886D-09, 7.64427040920545D-10,-2.75198005584737D-10,
     & 9.93792246090789D-11,-3.59877382902119D-11/
c
     DATA(A3(I), I=1,22) /  1.10642888011036D+01, 4.34299147561447D+00,
     &-2.33781774969295D-02, 4.23345215362947D-03, 8.68757084192089D-06,
     &-5.98261113270881D-04, 4.50490139240298D-04,-2.54858131942102D-04,
     & 1.27824189261340D-04,-5.97873878043957D-05, 2.66474012012582D-05,
     &-1.14381836209267D-05, 4.75393030377615D-06,-1.91759589929610D-06,
     & 7.50806465594834D-07,-2.84791180387123D-07, 1.04187791696225D-07,
     &-3.64567243689145D-08, 1.20129296139030D-08,-3.61030126779729D-09,
     & 9.12356140081759D-10,-1.36851363400914D-10/
c
     DATA C1, C2 / 2.35777520630369D-01, 1.35777520630369D+00/
c
     if (Y .ge. 0.5_RKind) then
       J = 1_IKind
       D = 1.0_RKind - Y
       D = D + D
     else if(Y .ge. 0.1_RKind) then
       J = 2_IKind
       D = 5.0_RKind * Y - 1.5_RKind
     else
       J = 3_IKind
       W = SQRT(-log(Y))
       D = C1 * W - C2
     endif
c
     TD  = D + D
     VNP1 = 0.0_RKind
     VN  = 0.0_RKind
     do K = 22_IKind, 2_IKind, -1_IKind
       TEMP = VN
       VN  = TD * VN - VNP1 + A(K, J)
       VNP1 = TEMP
     end do
     VN = D * VN - VNP1 + 0.5_RKind * A(1, J)
     if (J .eq. 1_IKind) VN = D * VN
     InvERFC = VN
     RETURN
     END FUNCTION InvERFC
c
c**********************************************************************
c
     integer(IKind) FUNCTION BiSearchOnv(R, FirstNdx, LastNdx)
c
```

```
c  Use bisection to find the minimum value that exceeds R in array v
c  between indexes FirstNdx and LastNdx, and return its index as the
c  function value. The logic assumes that:
c  1. The entries of array v are increasing
c  2. It is known that R is between the values given by the first and
c     last indexes
c  3. FirstNdx < LastNdx.
c  Input : R, FirstNdx, LastNdx, (module level array v)
c  Output: (function value)
c  Local : StartNdx, StopNdx, i
c
c  Arguments
c
      real(RKind) :: R
      integer(IKind) :: FirstNdx, LastNdx
c
c  Local variables
c
      integer(IKind) :: StartNdx, StopNdx, i
c
      StartNdx = FirstNdx
      StopNdx  = LastNdx
c
      Do While (StartNdx + 1_IKind .lt. StopNdx)
       i = StartNdx + Int((StopNdx - StartNdx) / 2_IKind)
       If(R .le. v(i)) then
         StopNdx  = i
       Else
         StartNdx = i
       End If
      end do
      BiSearchOnv = StopNdx
      RETURN
      END FUNCTION BiSearchOnv
c
c**********************************************************************
c
      real(RKind) FUNCTION FwdNorm(Z)
c
c  Numerically approximate the cumulative probability of being less than
c  deviate Z for the standard normal distribution, and return as the
c  function value. Use the (complementary) error function in Erf.
c  Input : Z
c  Output: function value
c  Local : ErfOpt, Root2, TwoRoot2, SixRoot2
c     .
```

```
c  Arguments
c
      real(RKind) :: Z
c
c  Local variables
c
      logical(LKind) :: ErfOpt
      real(RKind) :: Root2, TwoRoot2, SixRoot2
      DATA Root2, TwoRoot2, SixRoot2 / 1.414213562373095,
     &                    2.828427124746191,
     &                    8.485281374238572 /
c
c  Argument is tested for which interval it falls trying to avoid
c  excessive loss of significance.
c
      if(Z .gt. SixRoot2) then
        FwdNorm = 1.0_RKind
      else if(Z .lt. -SixRoot2) then
        FwdNorm = 0.0_RKind
      else if(Z .gt. TwoRoot2) then
        ErfOpt  = .false.
        FwdNorm = 1.0_RKind - 0.5_RKind * erf(ErfOpt, Z / Root2)
      else if(Z .lt. -TwoRoot2) then
        ErfOpt  = .false.
        FwdNorm = 0.5_RKind * erf(ErfOpt, -Z / Root2)
      else
        ErfOpt  = .true.
        FwdNorm = 0.5_RKind + 0.5_RKind * erf(ErfOpt, Z / Root2)
      end if
      RETURN
      END FUNCTION FwdNorm
c
c**********************************************************************
c
      real(RKind) FUNCTION Erf(ErfOpt, y)
c
c  Numerically approximate the (complementary) error function at Y using
c  Chebyshev expansions on one of 3 intervals and return as the function
c  value.  Algorithm due to Amos and Daniel.
c  If ErfOpt true, compute Erf(y), else compute Erfc(y) = 1 - Erf(y)
c  Input : ErfOpt, y
c  Output: function value
c  Local : x, ANS, XLIM
c
c  Arguments
c
```

```
      logical(LKind) :: ErfOpt
      real(RKind) :: y
c
c  Local variables
c
      real(RKind) :: x, ANS, XLIM
      DATA XLIM / 25.8408528684382 /
c
c  Argument out-of-bounds testing.
c
      if(ErfOpt) then
       if(y .le. -6.0_RKind) then
         Erf = -1.0_RKind
         RETURN
       else if(y .ge. 6.0_RKind) then
         Erf = 1.0_RKind
         RETURN
       end if
      else
       if(y .le. -6.0_RKind) then
         Erf = 2.0_RKind
         RETURN
       else if(y .ge. XLIM) then
         Erf = 0.0_RKind
         RETURN
       end if
      end if
c
c  Arguments must be positive.  If original negative, then answers are
c  adjusted afterwards.
c
      x = abs(y)
      if(x .lt. 2.0_RKind) then
       ANS = ErfAbsLT2(y)
       if(.not. ErfOpt) then
         ANS = 1.0_RKind - ANS
       end if
       Erf = ANS
       RETURN
      else if(x .lt. 4.0_RKind) then
       ANS = Erf2To4(x)
      else
       ANS = Erf4To6(x)
      end if
c
c  Here the absolute argument is between 2 and the upper limit.
```

```
c
      if(ErfOpt) then
       if(y .gt. 0.0_RKind) then
         ANS = 1.0_RKind - ANS
       else
         ANS = ANS - 1.0_RKind
       end if
      else
       if(y .lt. 0.0_RKind) then
         ANS = 2.0_RKind - ANS
       end if
      end if
      Erf = ANS
      RETURN
      END FUNCTION Erf
c
c*****************************************************************************
c
      real(RKind) FUNCTION ErfAbsLT2(X)
c
c  Numerically approximate the error function at X using Chebyshev
c  expansions for the interval -2 < X < 2 and return result as the
c  function value.
c  Input : X
c  Output: function value
c  Local : ANS, B1, B2, Z, A, N, I
c
c  Argument
c
      real(RKind) :: X
c
c  Local variables
c
      real(RKind) :: ANS, B1, B2, Z
      real(RKind), save :: A(31)
      integer(IKind) :: I, N
      DATA N / 31 /
      DATA(A(I), I = 1, 31) /
     & 2.96622112816961D+0,  0.0, -6.02142146773189D-1,  0.0,
     & 1.37989661379662D-1,  0.0, -2.78325425294437D-2,  0.0,
     & 4.84159904486783D-3,  0.0, -7.31727937169453D-4,  0.0,
     & 9.72419688637174D-5,  0.0, -1.14985131161804D-5,  0.0,
     & 1.22264871646933D-6,  0.0, -1.17982030973170D-7,  0.0,
     & 1.04140177691278D-8,  0.0, -8.46595329454225D-10, 0.0,
     & 6.37620443498960D-11, 0.0, -4.47177281962215D-12, 0.0,
     & 2.93540222982101D-13, 0.0, -1.83283038964141D-14 /
```

```
c
      Z = X / 2.0_RKind
      CALL ErfChebyPoly(N, A, Z, B1, B2)
      ANS = Z * (Z * B1 - B2 + A(1) / 2.0_RKind)
      ErfAbsLT2 = ANS
      RETURN
      END FUNCTION ErfAbsLT2
c
c*********************************************************************
c
      real(RKind) FUNCTION Erf2To4(X)
c
c  Numerically approximate the complementary error function at X using
c  Chebyshev expansions for the interval 2 < X < 4 and return as the
c  function value.
c  Input : X
c  Output: function value
c  Local : ANS, B1, B2, Z, A, N, I
c
c  Argument
c
      real(RKind) :: X
c
c  Local variables
c
      real(RKind) :: ANS, B1, B2, Z
      real(RKind), save :: A(16)
      integer(IKind) :: N, I
c
      DATA N / 16 /
      DATA(A(I), I = 1, 16) /
     & 1.06663088531993D+0,  1.78876062094436D-2, -3.80175293809401D-3,
     & 6.97111435023601D-4, -1.16368846063892D-4,  1.81367675932619D-5,
     &-2.67719939785138D-6,  3.77701329909996D-7, -5.12491142501402D-8,
     & 6.71870395763107D-9, -8.54019646112644D-10, 1.05544302186899D-10,
     &-1.27108990000124D-11, 1.49441348185064D-12,-1.71382907865335D-13,
     & 2.08899564313469D-14 /
c
      Z = X - 3.0_RKind
      CALL ErfChebyPoly(N, A, Z, B1, B2)
      ANS = Z * B1 - B2 + A(1) / 2.0_RKind
      ANS = EXP(-X * X) * ANS / X
      Erf2To4 = ANS
      RETURN
      END FUNCTION Erf2To4
c
```

```
c***********************************************************************
c
      real(RKind) FUNCTION Erf4To6(X)
c
c  Numerically approximate the complementary error function at X using
c  Chebyshev expansions for the interval 4 < X < 6 and return as the
c  function value.
c  Input : X
c  Output: function value
c  Local : ANS, B1, B2, Z, RTPI, A, N, I
c
c  Argument
c
      real(RKind) :: X
c
c  Local variables
c
      real(RKind) :: ANS, B1, B2, Z
      real(RKind), save :: A(27), RTPI
      integer(IKind) :: N, I
c
      DATA N / 27 /
      DATA RTPI / 1.7724538509552 /
      DATA(A(I), I = 1, 27) /
     & 1.97070527225754,    0.0, -1.43397402717750D-2, 0.0,
     & 2.97361692202619D-4, 0.0, -9.80351604336237D-6, 0.0,
     & 4.3313342034728D-7,  0.0, -2.362150026241D-8,   0.0,
     & 1.51549676581D-9,    0.0, -1.1084939856D-10,    0.0,
     & 9.04259014D-12,      0.0, -8.0947054D-13,       0.0,
     & 7.853856D-14,        0.0, -8.17918D-15,         0.0,
     & 9.0715D-16,          0.0, -1.0646D-16 /
c
      Z = 4.0_RKind / X
      CALL ErfChebyPoly(N, A, Z, B1, B2)
      ANS = Z * B1 - B2 + A(1) / 2.0_RKind
      ANS = (EXP(-X * X) / (X * RTPI)) * ANS
      Erf4To6 = ANS
      RETURN
      END FUNCTION Erf4To6
c
c***********************************************************************
c
      SUBROUTINE ErfChebyPoly(N, A, Z, B1, B2)
c
c  Evaluate a Chebyshev polynomial at Z using the N coefficients in A.
c  Return adjacent terms in B1 and B2.
```

```
c  Input : N, A, Z
c  Output: B1, B2
c  Local : I, TZ, S
c
c  Arguments
c
     real(RKind) :: A(*), Z, B1, B2
     integer(IKind) :: N
c
c  Local variables
c
     integer(IKind) :: I
     real(RKind) :: TZ, S
c
     TZ = Z + Z
     B1 = 0.0_RKind
     B2 = 0.0_RKind
     DO I = N, 2_IKind, -1_IKind
      S  = B1
      B1 = TZ * B1 - B2 + A(I)
      B2 = S
     end do
     RETURN
     END SUBROUTINE ErfChebyPoly

     END MODULE modstandardnormal
```

3. TESTFWDNORM

```
PROGRAM testfwdnorm
c
c  This program is designed for testing of the numerical approximation
c  to the forward normal. The test asks for a deviate and the code
c  responds with the cumulative probability.  Probabilities will be
c  written to screen, and optionally to file.
c
     Use modstandardnormal
     logical(LKind) :: WriteToFile
     real(RKind) :: p, z
c
     write(*, 9000)
 9000 format(' Save results to file (T/F)? ')
     read(*, *) WriteToFile
     if(WriteToFile) then
      open(10, file = 'TestFwdNorm.out')
      write(10, 9005)
```

```
 9005  format(10x,'z',22x,'p')
    end if
c
    write(*, 9010)
 9010 format(' Respond with a deviate outside (-8, 8) to stop.')
    z = 0.0
c
    do while (z .gt. -8.0 .and. z .lt. 8.0)
     write(*, 9020)
 9020  format(' Enter deviate (-8, 8) ')
     read(*, *) z
     if(z .le. -8.0 .or. z .ge. 8.0) then
      exit
     endif
c
     p = FwdNorm(z)
     write(*, 9050) z, p
 9050  format(1pe21.14,2x,1pe21.14)
     if(WriteToFile) then
      write(10, 9050) z, p
     end if
    end do
    if(WriteToFile) then
     close(10)
    end if

    END PROGRAM testfwdnorm
```

4. TESTINVNOR

```
PROGRAM testinvnor
c
c  This program is designed for testing of the numerical approximation
c  to the inverse normal. The test asks for a probability and the code
c  responds with the deviate.  Deviates will be written to screen, and
c  optionally to file.
c
    Use modstandardnormal
    logical(LKind) :: WriteToFile
    real(RKind) :: p, z
c
    write(*, 9000)
 9000 format(' Save results to file (T/F)? ')
    read(*, *) WriteToFile
    if(WriteToFile) then
     open(10, file = 'TestInvNor.out')
```

```
      write(10, 9005)
 9005  format(10x,'p',22x,'z')
    end if
c
    write(*, 9010)
 9010 format(' Respond with probability outside (0, 1) to stop.')
    p = 0.5
c
    do while (p .gt. 0. .and. p .lt. 1.)
     write(*, 9020)
 9020  format(' Enter probability (0, 1) ')
     read(*, *) p
     if(p .le. 0. .or. p .ge. 1.) then
      exit
     endif
c
     z = InvNor(p)
     write(*, 9050) p, z
 9050  format(1pe21.14,2x,1pe21.14)
     if(WriteToFile) then
      write(10, 9050) p, z
     end if
    end do
    if(WriteToFile) then
     close(10)
    end if

    END PROGRAM testinvnor
```

5. TESTNORMTABLE

```
PROGRAM testnormtable
c
c  This program is designed for testing of the interpolation of the
c  standard normal table. The test first asks for a probability and the
c  code responds with two deviates, 1 from InvNor and 1 from the table.
c  It then asks for a deviate and code responds with 2 probabilities,
c  1 from FwdNorm and 1 from the table.
c  Deviates and probabilities will be written to screen, and
c  optionally to file.
c
    Use modstandardnormal
    logical(LKind) :: WriteToFile
    real(RKind) :: p, p1, p2, z, z1, z2
c
    write(*, 9000)
```

```
9000 format(' Save results to file (T/F)? ')
   read(*, *) WriteToFile
   if(WriteToFile) then
    open(10, file = 'TestNormTable.out')
    write(10, 9005)
9005  format(10x,'p',18x,'InvNor-z',16x,'Table-z')
   end if
c
   write(*, 9010)
9010 format(' Respond with probability outside (0.01, 0.99) to stop.')
   p = 0.5
c
   do while (p .ge. 0.01 .and. p .le. 0.99)
    write(*, 9020)
9020  format(' Enter probability (0, 1) ')
    read(*, *) p
    if(p .lt. 0.01 .or. p .gt. 0.99) then
     exit
    endif
c
    z1 = InvNor(p)
    z2 = StdNorDev(p)
    write(*, 9050) p, z1, z2
9050  format(1pe21.14,2x,1pe21.14,2x,1pe21.14)
    if(WriteToFile) then
     write(10, 9050) p, z1, z2
    end if
   end do
c
   if(WriteToFile) then
    write(10, 9105)
9105  format(10x,'z',17x,'FwdNorm-p',16x,'Table-p')
   end if
c
   write(*, 9110)
9110 format(' Respond with a deviate outside (-2.33, 2.33) to stop.')
   z = 0.0
c
   do while (z .gt. -2.33 .and. z .lt. 2.33)
    write(*, 9120)
9120  format(' Enter deviate (-2.33, 2.33) ')
    read(*, *) z
    if(z .le. -2.33 .or. z .ge. 2.33) then
     exit
    endif
c
```

```
      p1 = FwdNorm(z)
      p2 = StdNorProb(z)
      write(*, 9150) z, p1, p2
 9150  format(1pe21.14,2x,1pe21.14,2x,1pe21.14)
      if(WriteToFile) then
        write(10, 9150) z, p1, p2
      end if
    end do
    if(WriteToFile) then
      close(10)
    end if

    END PROGRAM testnormtable
```

ATTACHMENT III

In this MathCad spreadsheet, the built in MathCad functions are used to evaluate the cumalative probability, p, for a deviate, x, and to evaluate the deviate, x, for a cumalative probability, p.

Input

$$x := \begin{bmatrix} -7.9 \\ -7.89 \\ -6.05 \\ -6 \\ -5 \\ -4 \\ -3 \\ -2.331 \\ -2.325 \\ -2 \\ -1 \\ 0 \\ 1 \\ 2 \\ 2.325 \\ 2.331 \\ 3 \\ 4 \\ 5 \\ 6.05 \\ 7 \\ 7.89 \\ 7.9 \end{bmatrix}$$

Given the deviate x, find the cumalative probability, p. This is called the Forward Normal.

Output:

pnorm(x,0,1) =

$1.39451714665928 \cdot 10^{-15}$
$1.51093268139137 \cdot 10^{-15}$
$7.24229170513766 \cdot 10^{-10}$
$9.86587645037701 \cdot 10^{-10}$
$2.86651571879195 \cdot 10^{-7}$
$3.16712418331200 \cdot 10^{-5}$
$1.34989803163010 \cdot 10^{-3}$
$9.87667984102212 \cdot 10^{-3}$
$1.00359801002741 \cdot 10^{-2}$
$2.27501319481792 \cdot 10^{-2}$
$1.58655253931457 \cdot 10^{-1}$
$5.00000000000000 \cdot 10^{-1}$
$8.41344746068543 \cdot 10^{-1}$
$9.77249868051821 \cdot 10^{-1}$
$9.89964019899726 \cdot 10^{-1}$
$9.90123320158978 \cdot 10^{-1}$
$9.98650101968370 \cdot 10^{-1}$
$9.99968328758167 \cdot 10^{-1}$
$9.99999713348428 \cdot 10^{-1}$
$9.99999999275771 \cdot 10^{-1}$
$9.99999999998720 \cdot 10^{-1}$
$9.99999999999998 \cdot 10^{-1}$
$9.99999999999999 \cdot 10^{-1}$

Excel Sheet (below) shows output from MathCad, and the two different Fortran algorithms (NormTable, Function and the error between the two and MathCad) .

a :=

A	B	C	D	E	F
Deviate (x)	MathCad (p)	NormTable (p)	Function (p)	Absolute Error |(B-C)|	Absolute Error |(B-D)|
-7.9	1.3945E-15	**N/A**	1.3945E-15	**N/A**	1.5556E-21
-7.89	1.5109E-15	**N/A**	1.5109E-15	**N/A**	1.6814E-21
-6.05	7.2423E-10	**N/A**	7.2423E-10	**N/A**	4.8778E-16
-6	9.8659E-10	**N/A**	9.8659E-10	**N/A**	6.5430E-16
-5	2.8665E-07	**N/A**	2.8665E-07	**N/A**	1.2722E-13
-4	3.1671E-05	**N/A**	3.1671E-05	**N/A**	9.1616E-12
-3	1.3499E-03	**N/A**	1.3499E-03	**N/A**	2.2754E-10
-2.331	9.8767E-03	**N/A**	9.8767E-03	**N/A**	1.0518E-09
-2.325	1.0036E-02	1.0036E-02	1.0036E-02	9.9036E-08	1.0638E-09
-2	2.2750E-02	2.2750E-02	2.2750E-02	4.4455E-08	1.8480E-09
-1	1.5866E-01	1.5866E-01	1.5866E-01	9.6537E-10	4.1412E-09
0	5.00E-01	5.0000E-01	5.0000E-01	0.0000E+00	0.0000E+00
1	8.41E-01	8.4134E-01	8.4134E-01	9.6537E-10	4.1412E-09
2	9.77E-01	9.7725E-01	9.7725E-01	4.4455E-08	1.8480E-09
2.325	9.90E-01	9.8996E-01	9.8996E-01	9.9036E-08	1.0638E-09
2.331	9.90E-01	**N/A**	9.9012E-01	**N/A**	1.0518E-09
3	9.99E-01	**N/A**	9.9865E-01	**N/A**	2.2754E-10
4	1.00E+00	**N/A**	1.00E+00	**N/A**	9.1630E-12
5	1.00E+00	**N/A**	1.00E+00	**N/A**	1.2701E-13
6.05	1.00E+00	**N/A**	1.00E+00	**N/A**	9.9920E-16
7	1.00E+00	**N/A**	1.00E+00	**N/A**	0.0000E+00
7.89	1.00E+00	**N/A**	1.00E+00	**N/A**	0.0000E+00
7.9	1.00E+00		1.00E+00		9.9920E-16

Given the cumalative probability p, find the deviate x
This is called the Inverse Normal.

$$p := \begin{bmatrix} .000000000000001 \\ .002 \\ .004 \\ .008 \\ .010 \\ .01005 \\ .01007 \\ .01009 \\ .02 \\ .06 \\ .1 \\ .2 \\ .4 \\ .5 \\ .7 \\ .999999999999999 \end{bmatrix}$$

$$qnorm(p,0,1) = \begin{bmatrix} -7.94140555777135 \\ -2.87816173909549 \\ -2.6520698079022 \\ -2.40891554581546 \\ -2.32634787404084 \\ -2.32447593301054 \\ -2.32372943172158 \\ -2.3229842231219 \\ -2.05374891063183 \\ -1.55477359459685 \\ -1.2815515655446 \\ -0.841621233572915 \\ -0.253347103135799 \\ 0 \\ 0.52440051270804 \\ 7.94100472087549 \end{bmatrix}$$

Excel Sheet shows output from
MathCad, and the two different
Fortran algorithms (NormTable and
and the error between the two.

b :=

A	B	C	D	E	F
Cum Prob. (p)	MathCad (x)	NormTable (x)	Function (x)	Absolute Error |(B-C)|	Absolute Error |(B-D)|
1.0000E-15	-7.9414E+00	**N/A**	-7.9420E+00	**N/A**	5.9444E-04
2.0000E-03	-2.8782E+00	**N/A**	-2.8782E+00	**N/A**	4.9370E-08
4.0000E-03	-2.6521E+00	**N/A**	-2.6521E+00	**N/A**	4.5278E-08
8.0000E-03	-2.4089E+00	**N/A**	-2.4089E+00	**N/A**	4.1107E-08
1.0000E-02	-2.3263E+00	-2.3263E+00	-2.3263E+00	3.9805E-08	3.9805E-08
1.0050E-02	-2.3245E+00	-2.3245E+00	-2.3245E+00	4.0153E-06	3.9775E-08
1.0070E-02	-2.3237E+00	-2.3237E+00	-2.3237E+00	3.3622E-06	3.9764E-08
1.0090E-02	-2.3230E+00	-2.3230E+00	-2.3230E+00	1.4164E-06	3.9752E-08
2.0000E-02	-2.0537E+00	-2.0537E+00	-2.0537E+00	3.5281E-08	3.5281E-08
6.0000E-02	-1.5548E+00	-1.5548E+00	-1.5548E+00	2.6589E-08	2.6589E-08
1.0000E-01	-1.2816E+00	-1.2816E+00	-1.2816E+00	2.1930E-08	2.1930E-08
2.0000E-01	-8.4162E-01	-8.4162E-01	-8.4162E-01	1.4393E-08	1.4393E-08
4.0000E-01	-2.5335E-01	-2.5335E-01	-2.5335E-01	4.3358E-09	4.3359E-09
5.0000E-01	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
7.0000E-01	5.2440E-01	5.2440E-01	5.2440E-01	8.9747E-09	8.9747E-09
1.0000E+00	7.94100472		7.94E+00		4.3963E-04

ATTACHMENT IV
FORTRAN OUTPUT FILES

TestFwdNorm.out:

z	p
-7.90000000000000E+00	1.39451559106967E-15
-7.89000000000000E+00	1.51093100002460E-15
-6.05000000000000E+00	7.24228682733315E-10
-6.00000000000000E+00	9.86586990739334E-10
-5.00000000000000E+00	2.86651444658299E-07
-4.00000000000000E+00	3.16712326714731E-05
-3.00000000000000E+00	1.34989780408652E-03
-2.33100000000000E+00	9.87667878923520E-03
-2.32500000000000E+00	1.00359790364380E-02
-2.00000000000000E+00	2.27501301001457E-02
-1.00000000000000E+00	1.58655249790304E-01
0.00000000000000E+00	5.00000000000000E-01
1.00000000000000E+00	8.41344750209696E-01
2.00000000000000E+00	9.77249869899854E-01
2.32500000000000E+00	9.89964020963562E-01
2.33100000000000E+00	9.90123321210765E-01
3.00000000000000E+00	9.98650102195913E-01
4.00000000000000E+00	9.99968328767329E-01
5.00000000000000E+00	9.99999713348555E-01
6.05000000000000E+00	9.99999999275771E-01
7.00000000000000E+00	9.99999999998720E-01
7.89000000000000E+00	9.99999999999998E-01
7.90000000000000E+00	9.99999999999999E-01

TestInvNor.out:

p	z
1.00000000000000E-15	-7.94200000000000E+00
2.00000000000000E-03	-2.87816168972574E+00
4.00000000000000E-03	-2.65206976262383E+00
8.00000000000000E-03	-2.40891550470837E+00
1.00000000000000E-02	-2.32634783423627E+00
1.00500000000000E-02	-2.32447589323511E+00
1.00700000000000E-02	-2.32372939195777E+00
1.00900000000000E-02	-2.32298418336969E+00
2.00000000000000E-02	-2.05374887535086E+00
6.00000000000000E-02	-1.55477356800796E+00

1.00000000000000E-01	-1.28155154361505E+00
2.00000000000000E-01	-8.41621219179845E-01
4.00000000000000E-01	-2.53347098799945E-01
5.00000000000000E-01	0.00000000000000E+00
7.00000000000000E-01	5.24400503733308E-01
9.99999999999999E-01	7.94144435150413E+00

TestNormTable.out:

p	InvNor-z	Table-z
1.00000000000000E-02	-2.32634783423627E+00	-2.32634783423627E+00
1.00500000000000E-02	-2.32447589323511E+00	-2.32447994827188E+00
1.00700000000000E-02	-2.32372939195777E+00	-2.32373279388612E+00
1.00900000000000E-02	-2.32298418336969E+00	-2.32298563950037E+00
2.00000000000000E-02	-2.05374887535086E+00	-2.05374887535086E+00
6.00000000000000E-02	-1.55477356800796E+00	-1.55477356800796E+00
1.00000000000000E-01	-1.28155154361505E+00	-1.28155154361504E+00
2.00000000000000E-01	-8.41621219179845E-01	-8.41621219179865E-01
4.00000000000000E-01	-2.53347098799945E-01	-2.53347098800017E-01
5.00000000000000E-01	0.00000000000000E+00	0.00000000000000E+00
7.00000000000000E-01	5.24400503733308E-01	5.24400503733356E-01

z	FwdNorm-p	Table-p
-2.32500000000000E+00	1.00359790364380E-02	1.00360791360384E-02
-2.00000000000000E+00	2.27501301001457E-02	2.27501764027591E-02
-1.00000000000000E+00	1.58655249790304E-01	1.58655254896831E-01
0.00000000000000E+00	5.00000000000000E-01	5.00000000000000E-01
1.00000000000000E+00	8.41344750209696E-01	8.41344745103169E-01
2.00000000000000E+00	9.77249869899854E-01	9.77249823597241E-01
2.32500000000000E+00	9.89964020963562E-01	9.89963920863962E-01